

ENKBG-1.0

VAX 11/730 APT/RD  
LOOPBACK TST

EP-ENKBG-DL-1.0  
FICHE 1 OF 1

JUL 1982  
COPYRIGHT© 1982  
MADE IN USA

disi20

IDENTIFICATION

Product code: ENKBG VERSION 1.00  
Product name: VAX-11/730 APT/RD USART MODEM LOOPBACK TEST  
Product date: 30-APR-1982  
Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC  
DECUS  
UNIBUS

DECsystem-10  
MASSBUS  
VAX

DECSYSTEM-20  
PDP  
VMS

digital

Table of Contents

|     |                                                |   |
|-----|------------------------------------------------|---|
| 1.0 | ABSTRACT . . . . .                             | 3 |
| 2.0 | HARDWARE REQUIREMENTS . . . . .                | 3 |
| 3.0 | SOFTWARE REQUIREMENTS . . . . .                | 3 |
| 4.0 | PREREQUISITES . . . . .                        | 3 |
| 5.0 | OPERATING INSTRUCTIONS . . . . .               | 4 |
| 5.1 | Options . . . . .                              | 4 |
| 5.2 | Event Flags . . . . .                          | 4 |
| 5.3 | APT Setup . . . . .                            | 4 |
| 6.0 | PROGRAM FUNCTIONAL DESCRIPTION . . . . .       | 4 |
| 6.1 | Program Overview . . . . .                     | 4 |
| 6.2 | Program Size . . . . .                         | 4 |
| 6.3 | Program Run Times . . . . .                    | 5 |
| 6.4 | Run-time Dynamics . . . . .                    | 5 |
| 6.5 | Fault Detection . . . . .                      | 5 |
| 6.6 | Performance During Hardware Failures . . . . . | 5 |
| 6.7 | Program Applications . . . . .                 | 6 |
| 6.8 | Test Descriptions . . . . .                    | 6 |
| 7.0 | MAINTENANCE HISTORY . . . . .                  | 6 |

## 1.0 ABSTRACT

This program is an 8085 based micro-diagnostic designed to test the APT/RD usart hardware on the WCS (Writable Control Store) module of the VAX-11/730 processor.

## 2.0 HARDWARE REQUIREMENTS

This program will run with a minimum hardware configuration of the WCS and CPU modules present. It also requires the use of either a modem in loopback mode or a turnaround connector on the APT/RD Usart.

Other hardware modules and options may be installed without affecting the diagnostic's performance.

## 3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into the top 3K of 8085 RAM. The micro-diagnostic monitor also resides in 8085 ram, where console software is normally present.

## 4.0 PREREQUISITES

This module should be run after running modules ENKBA thru ENKBD. It thereby assumes that the Usart internal loopback selftests have successfully completed.

This program also assumes that there is some sort of wraparound capability enabled on the APT-RD USART. This could either mean a modem set in wraparound mode, or a turnaround connector attached to the port cable.

## 5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKBG after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

### 5.1 Options

Not applicable

### 5.2 Event Flags

Not applicable

### 5.3 APT Setup

Unlike other VAX-11/730 micro-diagnostics, ENKBG can not be run under APT. Though the program may be downline loaded through the APT-RD port like other micro-diagnostics, because it requires some kind of turnaround on the port, it can never be run successfully under APT.

## 6.0 PROGRAM FUNCTIONAL DESCRIPTION

### 6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. Though the APT-RD USART drivers and receivers are the most likely sources of error, this program will also detect faults in the cable, connectors, and the inversion logic that is between the APT-RD USART and the connector.

### 6.2 Program Size

This program runs in 8085 RAM memory and is approximately 512 bytes long.

### 6.3 Program Run Times

This program will take approximately 10 seconds to run.

### 6.4 Run-time Dynamics

Not applicable

### 6.5 Fault Detection

The diagnostic will report errors with the following header:

SECT TST ERR EXP REC OTHER MSK MODULE

1. SECT is the program name (ENKBG)
2. TST is the test number that failed. (Test 2F is the only test.)
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

### 6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system (if a battery backup is not present). Other hardware failures will cause normal error printouts.

## 6.7 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is NOT APT compatible.

Customers may use the program to verify the basic integrity of the APT-RD port on the WCS module in the system.

## 6.8 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

## 7.0 MAINTENANCE HISTORY

| DATE      | VERSION | DESCRIPTION      |
|-----------|---------|------------------|
| 30-APR-82 | 1.0     | Initial release. |

ZZ-ENKBG-1.0 7000 4

: TITLE 'ENKBG VAX-11/730 APT/RD USART MODEM LOOPBACK TEST Rev 01.00'

: COPYRIGHT (c) 1982 BY  
: DIGITAL EQUIPMENT CORPORATION, MAYNARD,  
: MASSACHUSETTS. ALL RIGHTS RESERVED.: THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
: ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION  
: OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER COPIES THEREOF  
: MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON. NO  
: TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.: THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND  
: SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.: DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
: SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.: ++  
: FACILITY: VAX-11/730 MICRO-DIAGNOSTIC.: ABSTRACT: This 8085 based diagnostic tests the hardcore logic of the  
: APT/RD USART of the WCS module.

: ENVIRONMENT: ENKAA Micro-diagnostic Monitor

: AUTHOR: David Mayo 27-APR-1982 VERSION 01.00

: MODIFIED BY:  
: --7000 3  
7000 4  
7000 5  
7000 6  
7000 7  
7000 8  
7000 9  
7000 10  
7000 11  
7000 12  
7000 13  
7000 14  
7000 15  
7000 16  
7000 17  
7000 18  
7000 19  
7000 20  
7000 21  
7000 22  
7000 23  
7000 24  
7000 25  
7000 26  
7000 27  
7000 28  
7000 29  
7000 30  
7000 31  
7000 32  
7000 33  
7000 39  
7000 40  
7000 41  
7000 42  
7000 43  
7000 44  
7000 45  
7000 46  
7000 47  
7000 48  
7000 49  
7000 50  
7000 51  
7000 52  
7000 53  
7000 54  
7000 55  
7000 56  
7000 57  
7000 58  
7000 59  
7000 60  
7000 61  
7000 62  
7000 63  
7000 64  
7000 65  
7000 66  
7000 67

ZZ-ENKBG-1.0 7000 4

.PSECT CPU6,BYTE  
.ALIGN BYTE

```

*****
: DATA AREA
: *****

```

```

7000 68
7000 69 000
7000 70
7000 71
7000 72
7000 73
7000 74
7000 75
7000 76
7000 77
7000 78 00000000
7000 79
7000 80 00000000
      00000002
      00000004
      00000006
      00000006
      00000001
      00000003
      00000005
      00000007 00000006
7000 81
7000 82 00000000
7000 83
7000 84
7000 85 00000001
7000 86 00000001
7000 87
7000 88
7000 89 00000002
7000 90
7000 91 00000002
7000 92
7000 93
7000 94 00000003
7000 95
7000 96 00000003
7000 97
7000 98
7000 99 00000004
7000 100 00000004
7000 101
7000 102 00000005
7000 103
7000 104 00000006
7000 105
7000 106 00000007
7000 107
7000 108 00000007
7000 109
7000 110
7000 111 00000008
7000 112 00000009
7000 113 0000000A
7000 114
7000 115 0000001C
7000 116 0000001D
7000 117
7000 118 00000020
7000 119 00000021

```

HEAD =0

EQUATE

;REGISTER EQUATE MACRO

UPC14A

=&lt;^X00&gt;

;(RO) UPC BIT 14 ASSERTED HIGH &lt;MSB&gt;

MISC2

=&lt;^X01&gt;

BOOTF

=&lt;^X01&gt;

;(RO) BOOT FLAG. ASSERTED LOW. &lt;LSB&gt;

DCLO

=&lt;^X02&gt;

;(RO) DC LOW FLAG ASSERTED HIGH&lt;MSB&gt;

HLTFLG

=&lt;^X02&gt;

;(RO) HALT FLAG ASSERTED LOW &lt;LSB&gt;

READJ1

=&lt;^X03&gt;

;(RO) JUMPER J1 FLAG ASS. HIGH &lt;MSB&gt;

ALLOWP

=&lt;^X03&gt;

;(RO) ENABLE CONTROL P ASS LOW &lt;LSB&gt;

READJ2

=&lt;^X04&gt;

APTFLG

=&lt;^X04&gt;

;(RO) BIT 07 EQUALS J2 &lt;MSB&gt;

;(RO) APT PRESENT FLAG &lt;LSB&gt;

AUTO

=&lt;^X05&gt;

;(RO) AUTO TEST FLAG ASS LOW &lt;LSB&gt;

REMDIS

=&lt;^X06&gt;

;(RO) REMOTE DISABLE FLAG ASS LOW &lt;LSB&gt;

OKV15

=&lt;^X07&gt;

;(RO) 15 VOLTS OK FLAG ASS LOW &lt;MSB&gt;

BOOTEN

=&lt;^X07&gt;

;(RO) BOOT ENABLE FLAG ASS LOW &lt;LSB&gt;

WCSB0

=&lt;^X08&gt;

WCSB1

=&lt;^X09&gt;

WCSB2

=&lt;^X0A&gt;

;(WO) WCS WRITE DATA REGISTER

ITDB

=&lt;^X1C&gt;

ITCSR

=&lt;^X1D&gt;

;(R/W) TIMER DATA REGISTER

;(R/W) TIMER STATUS REGISTER

CLRCLK

=&lt;^X20&gt;

SETCLK

=&lt;^X21&gt;

;(WO) CLEAR CPU CLOCK RUN

;(WO) SET CPU CLOCK RUN

ZZ-ENKBG-1.0 7000 4  
 7000 120  
 7000 121 00000022  
 7000 122 00000023  
 7000 123  
 7000 124 00000024  
 7000 125 00000025  
 7000 126  
 7000 127 00000026  
 7000 128 00000027  
 7000 129  
 7000 130 00000028  
 7000 131 00000029  
 7000 132  
 7000 133 0000002A  
 7000 134 0000002B  
 7000 135  
 7000 136 0000002C  
 7000 137 0000002D  
 7000 138  
 7000 139 0000002E  
 7000 140 0000002F  
 7000 141  
 7000 142 00000030  
 7000 143 00000031  
 7000 144  
 7000 145 00000032  
 7000 146 00000033  
 7000 147  
 7000 148 00000034  
 7000 149 00000035  
 7000 150  
 7000 151 00000036  
 7000 152 00000037  
 7000 153  
 7000 154 00000038  
 7000 155 00000039  
 7000 156  
 7000 157 0000003C  
 7000 158 0000003D  
 7000 159  
 7000 160 00000040  
 7000 161 00000041  
 7000 162 00000042  
 7000 163  
 7000 164  
 7000 165  
 7000 166 00000043  
 7000 167  
 7000 168 00000044  
 7000 169 00000045  
 7000 170 00000046  
 7000 171  
 7000 172  
 7000 173  
 7000 174 00000047  
 7000 175  
 7000 176 00000048  
 7000 177 00000049  
 7000 178 0000004A  
 7000 179

CLRSS =<^X22>  
 SETSS =<^X23>  
 CLRCCK =<^X24>  
 SETCCK =<^X25>  
 CLRUPK =<^X26>  
 SETUPK =<^X27>  
 SETMCI =<^X28>  
 CLRMCI =<^X29>  
 SETMCK =<^X2A>  
 CLRMCK =<^X2B>  
 CLRTMR =<^X2C>  
 SETTMR =<^X2D>  
 CLRBSY =<^X2E>  
 SETBSY =<^X2F>  
 SETDCL =<^X30>  
 CLRDCL =<^X31>  
 SETACL =<^X32>  
 CLRACL =<^X33>  
 INITL =<^X34>  
 INITH =<^X35>  
 CLRWCK =<^X36>  
 SETWCK =<^X37>  
 CLRCCK =<^X38>  
 SETCCK =<^X39>  
 CLRLIT =<^X3C>  
 SETLIT =<^X3D>  
 TRDB =<^X40>  
 TRSR =<^X41>  
 TRMODE =<^X42>  
 TRCR =<^X43>  
 TUDB =<^X44>  
 TUSR =<^X45>  
 TUMODE =<^X46>  
 TUCR =<^X47>  
 TTDB =<^X48>  
 TTSR =<^X49>  
 TTMODE =<^X4A>

;(WO) CLEAR CLOCK SINGLE STEP  
 ;(WO) SET CLOCK SINGLE STEP  
 ;(WO) CLEAR CSR CLOCK  
 ;(WO) SET CSR CLOCK  
 ;(WO) CLEAR THE UPC CLOCK  
 ;(WO) SET THE UPC CLOCK  
 ;(WO) SET MEMORY CONTROL INIT  
 ;(WO) CLEAR MEMORY CONTROL INIT  
 ;(WO) SET MEMORY CONTROL CLOCK  
 ;(WO) CLEAR MEMORY CONTROL CLOCK  
 ;(WO) STOP TIMER  
 ;(WO) START TIMER COUNTING  
 ;(WO) CLEAR UNIBUS BUS BUSY  
 ;(WO) SET UNIBUS BUS BUSY  
 ;(WO) SET DC LOW  
 ;(WO) CLEAR DC LOW  
 ;(WO) SET AC LOW  
 ;(WO) CLEAR AC LOW  
 ;(WO) CLEAR UNIBUS INIT  
 ;(WO) SET UNIBUS INIT  
 ;(WO) CLEAR WCS BIT 00  
 ;(WO) SET WCS BIT 00  
 ;(WO) CLEAR WCS BIT 01  
 ;(WO) SET WCS BIT 01  
 ;(WO) CLEAR THE RUN LIGHT  
 ;(WO) SET THE RUN LIGHT  
 ;(R/W) REMOTE USART DATA BUFFER  
 ;(RO) REMOTE USART STATUS REGISTER  
 ;(R/W) REMOTE MODE REGISTER 1 + 2  
 ;FIRST ACCESS GET MR1, SECOND GETS MR2  
 ;READING THE CR FORCES THE INTERNAL  
 ;POINTER BACK TO MR1  
 ;(R/W) REMOTE USART COMMAND REGISTER  
 ;(R/W) TU-58 DATA BUFFER  
 ;(RO) TU-58 STATUS REGISTER  
 ;(R/W) TU-58 MODE REGISTER 1 + 2  
 ;FIRST ACCESS GET MR1, SECOND GETS MR2  
 ;READING THE CR FORCES THE INTERNAL  
 ;POINTER BACK TO MR1  
 ;(R/W) TU-58 COMMAND REGISTER  
 ;(R/W) TERMINAL DATA BUFFER  
 ;(RO) TERMINAL STATUS REGISTER  
 ;(R/W) TERMINAL MODE REGISTER 1 + 2  
 ;FIRST ACCESS GET MR1, SECOND GETS MR2

ZZ-ENKBG-1.0    7000    4  
 7000 180  
 7000 181  
 7000 182 0000004B  
 7000 183  
 7000 184 00000080  
 7000 185  
 7000 186 00000082  
 7000 187  
 7000 188 00000083  
 7000 189  
 7000 190 00000084  
 7000 191  
 7000 192 00000085  
 7000 193  
 7000 194 00000086  
 7000 195  
 7000 196 00000087  
 7000 197  
 7000 198 000000A0  
 7000 199 000000A1  
 7000 200  
 7000 201 000000A2  
 7000 202 000000A3  
 7000 203  
 7000 204 000000A4  
 7000 205 000000A5  
 7000 206  
 7000 207 000000A6  
 7000 208 000000A7  
 7000 209  
 7000 210 000000A8  
 7000 211 000000A9  
 7000 212  
 7000 213  
 7000 214  
 7000 215  
 7000 216  
 7000 217 000000A8  
 7000 218 000000A9  
 7000 219  
 7000 220 000000AA  
 7000 221 000000AB  
 7000 222  
 7000 223 000000AC  
 7000 224 000000AD  
 7000 225  
 7000 226 000000AE  
 7000 227 000000AF  
 7000 228  
 7000 229 000000CC  
 7000 230  
 7000 231 000000EC  
 7000 232  
 7000 233  
 7000 234  
 7000 235  
 7000 236 00004C22  
 7000 237 00004C25  
 7000 238  
 7000 239

K 1

Fiche 1    Frame K1    Sequence 10  
 ;READING THE CR FORCES THE INTERNAL  
 ;POINTER BACK TO MR1  
 ;(R/W) TERMINAL COMMAND REGISTER  
 ;(RO) CONSOLE READ REGISTER  
 ;(RO) ACKNOWLEDGE FROM CPU    <LSB>  
 ;(RO) ATTENTION FROM CPU    <LSB>  
 ;(RO) UPC BIT 14    <LSB>  
 ;(RO) CSR BIT 7    <LSB>  
 ;(RO) CSR BIT 15    <LSB>  
 ;(RO) CSR BIT 23    <LSB>  
 ;(WO) ENABLE STALL ON PARITY ERROR  
 ;(WO) DISABLE STALL ON PARITY ERROR  
 ;(WO) SET THE V-BUS TO SHIFT MODE  
 ;(WO) SET THE V-BUS TO NORMAL MODE  
 ;(WO) SET THE INTERVAL TIMER INTERRUPT  
 ;(WO) CLEAR THE INTERVAL TIMER INTRPT.  
 ;(WO) ENABLE MEMORY REFERENCES  
 ;(WO) DISABLE MEMORY REFERENCES  
 ;(WO) SET CONSOLE ACKNOWLEDGE  
 ;(WO) CLEAR CONSOLE ACKNOWLEDGE  
 ;NOTE: THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC  
 ;SERIAL INPUT. WHEN USED THIS WAY THE SENSE IS INVERTED. TO SIMPLIFY  
 ;MATTERS THE NEXT TWO EQUATES ARE INCLUDED  
 ;(WO) CLEAR THE V-BUS SERIAL INPUT  
 ;(WO) SET THE V-BUS SERIAL INPUT  
 ;(WO) SET CONSOLE ATTENTION  
 ;(WO) CLEAR CONSOLE ATTENTION  
 ;(WO) SET POWER FAIL INTERRUPT  
 ;(WO) CLEAR POWER FAIL INTERRUPT  
 ;(WO) SET THE CONSOLE HALT BIT  
 ;(WO) CLEAR THE CONSOLE HALT BIT  
 ;(WO) THE CONSOLE WRITE REGISTER  
 ;(RO) Y-BUS READ  
 ;ADDITIONAL DATA  
 PARITY\_VECTOR    = <^X4C22>    ;PARITY ERROR BRANCH VECTOR.  
 POWER\_VECTOR    = <^X4C25>    ;POWER FAIL BRANCH VECTOR.  
 ;USART DATA

TTCR    = <^X4B>  
 READ    = <^X80>  
 CPUACK    = <^X82>  
 CPATTN    = <^X83>  
 UPC14    = <^X84>  
 CSR07    = <^X85>  
 CSR15    = <^X86>  
 CSR23    = <^X87>  
 ENBPAR    = <^XA0>  
 DISPAR    = <^XA1>  
 VSHIFT    = <^XA2>  
 VNORML    = <^XA3>  
 SETTIM    = <^XA4>  
 CLRTIM    = <^XA5>  
 ENBMEM    = <^XA6>  
 DISMEM    = <^XA7>  
 SETACK    = <^XA8>  
 CLRACT    = <^XA9>  
 CLRUPC    = <^XA8>  
 SETUPC    = <^XA9>  
 SETATN    = <^XAA>  
 CLRACTN    = <^XAB>  
 SETPFI    = <^XAC>  
 CLRPF I    = <^XAD>  
 SETHLT    = <^XAE>  
 CLRHLT    = <^XAF>  
 WRITE    = <^XCC>  
 YBUSRD    = <^XEC>

;ADDITIONAL DATA

PARITY\_VECTOR    = <^X4C22>  
 POWER\_VECTOR    = <^X4C25>

;USART DATA

```

ZZ-ENKBG-1.0    7000    4
7000 240
7000 241 00000002
7000 242 00000037
7000 243
7000 244
7000 245
7000 246
7000 247
7000 248
7000 249
7000 250
7000 251 00CC'C3
7003 252
7003 253
7003 254
7003 255
7003 256
7003 257
7003 258
7003 259
7003 260 0100
7005 261 47 42
7007 262
7007 263
7007 264
7007 265
7007 266
7007 267
7007 268
7007 269 00
7008 270 00
7009 271 00
700A 272 FF
700B 273 00
700C 274 00
700D 275
700D 276
700D 277
700D 278
700D 279
700D 280
700D 281
700D 282
700D 283
700D 284
700D 285 0009'32
7010 286
7010 287 A8 D3
7012 288 A9 D3
7014 289
7014 290 41 DB
7016 291 1F
7017 292 0038'DA
701A 293
701A 294 00 FF 01
701D 295 0B
701E 296 78
701F 297 B1
7020 298 001D'C2
7023 299

```

L 1

Fiche 1 Frame L1

Sequence 11

```

TRSR_MASK      = <^X02>          ;MASK TO READ TR STATUS REGISTER
TRCR_MASK      = <^X37>          ;MASK TO SET UP TR COMMAND REGISTER

```

```

*****
: PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
:
*****

```

```

ENTRY:          JMP      TEST2F

```

```

*****
: VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME.  THESE 2 WORDS CAN NOT
: BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
:
*****

```

```

VERSION:         .WORD      <^X0100>      ;VERSION NUMBER (REV 01.00)
FILE_NAME:       .ASCII    'BG'          ;LAST 2 CHARS OF FILE NAME

```

```

*****
: VARIABLES AREA
:
*****

```

```

BACKGRND:       .BYTE      0              ;BACKGROUND OF 1'S OR 0'S
DATA_PAT:       .BYTE      0              ;SHIFTING DATA PATTERN
EXP_DAT:        .BYTE      0              ;EXPECTED DATA AREA
ONE_DAT:        .BYTE      <^XFF>        ;DATA PATTERN OF ALL ONES
REC_DAT:        .BYTE      0              ;RECEIVED DATA AREA
ZERO_DAT:       .BYTE      0              ;DATA PATTERN OF ALL ZEROS

```

```

*****
: TEST2F_SUB  THIS ROUTINE IS USED BY TEST 2F TO WRITE AN 8 BIT TEST PATTERN
: TO THE APT/RD UART AND THEN READ IT BACK
:
*****

```

```

TEST2F_SUB:     STA      EXP_DAT          ;LOAD THE DATA PATTERN INTO EXP_DAT
1$:             OUT      SETACK           ;SET THE SCOPE SYNC
               OUT      CLRACK           ;CLEAR IT
               IN       TRSR             ;LOOK FOR XMIT RDY
               RAR
               JC        3$              ;CONTINUE IF XMIT RDY SET
2$:             LXI      B,<^X00FF>       ;WAIT HERE
               DCX      B
               MOV      A,B
               ORA      C
               JNZ      2$

```

```

ZZ-ENKBG-1.0 7000 4
7023 300 41 DB
7025 301 1F
7026 302 0038'DA
7029 303
7029 304 01 3E
702B 305 0000'32
702E 306 0083'CD
7031 307
7031 308 0000'3A
7034 309 0F
7035 310 0010'DA
7038 311
7038 312 0009'3A
703B 313
703B 314 40 D3
703D 315
703D 316 02 06
703F 317
703F 318 41 DB
7041 319 A0
7042 320 0065'C2
7045 321
7045 322 3F FF 01
7048 323 0B
7049 324 78
704A 325 B1
704B 326 0048'C2
704E 327
704E 328 02 06
7050 329 41 DB
7052 330 A0
7053 331 0065'C2
7056 332
7056 333 02 3E
7058 334 0000'32
705B 335 0083'CD
705E 336
705E 337 0000'3A
7061 338 0F
7062 339 0010'DA
7065 340
7065 341 40 DB
7067 342 000B'32
706A 343
706A 344 00'09'21
01 0E 00'0B'11
007B'CA 0000'CD
7078 345
7078 346
7078 347 009B'CD
707B 348
707B 349
707B 350
707B 351 0000'3A
707E 352 0F
707F 353 0010'DA
7082 354
7082 355 C9
7083 356
7083 357

```

```

*****
*
*
*
*
*****

```

```

M 1
IN TRSR
RAR
JC 3$
MVI A,1
STA CON_ERR_NUM
CALL TEST2F_ERR1_2
LDA ERROR_CON
RRC
JC 1$
LDA EXP_DAT
OUT TRDB
MVI B,TRSR_MASK
IN TRSR
ANA B
JNZ 5$
LXI B,<^X3FFF>
DCX B
MOV A,B
ORA C
JNZ 4$
MVI B,TRSR_MASK
IN TRSR
ANA B
JNZ 5$
MVI A,2
STA CON_ERR_NUM
CALL TEST2F_ERR1_2
LDA ERROR_CON
RRC
JC 1$
IN TRDB
STA REC_DAT
IFNEQ EXP_DAT,REC_DAT,1
CALL TEST2F_ERROR
ENDIF
LDA ERROR_CON
RRC
JC 1$
RET

```

Fiche 1 Frame M1 Sequence 12

;THEN LOOK FOR XMIT RDY AGAIN

;CONTINUE IF XMIT RDY SET

;ELSE REPORT ERROR 1

;PRINT AN ERROR MESSAGE

;IF THE ERROR LOOPING FLAG IS SET...

;...JUMP TO BEGINNING OF ERROR LOOP

;GET THE PATTERN

;SEND IT OUT

;GET THE SR

;MASK OUT ALL BITS EXCEPT REVR DONE

;CONTINUE IF REVR DONE SET

;WAIT HERE

;THEN LOOK FOR REVR DONE AGAIN

;CONTINUE IF REVR DONE SET

;ELSE REPORT ERROR 2

;PRINT AN ERROR MESSAGE

;IF THE ERROR LOOPING FLAG IS SET...

;...JUMP TO BEGINNING OF ERROR LOOP

;GET THE DATA

;STORE DATA IN REC\_DAT

; DATA

;PRINT AN ERROR MESSAGE

;IF THE ERROR LOOPING FLAG IS SET...

;...JUMP TO BEGINNING OF ERROR LOOP

```

ZZ-ENKBG-1.0      7000      4
7083 358
7083 359
7083 360
7083 361
7083 362
7083 363
7083 364
7083 365 0000'32 3C AF
7088 366 0009'3A
708B 367 0003'32
708E 368 0000'32 AF
7092 369 00 00 21
7095 370 0000'22
7098 371 00BD'C3
709B 372
709B 373 03 3E
709D 374 0000'32
70A0 375
70A0 376 0009'3A
70A3 377 0003'32
70A6 378
70A6 379 000B'3A
70A9 380 0003'32
70AC 381
70AC 382 0000'32 3C AF
70B1 383
70B1 384 FF FF 21
70B4 385 0000'22
70B7 386 00 FF 21
70BA 387 0002'22
70BD 388
70BD 389 00'00'21
70C0 390
70C0 391 00'00'11
70C3 392 05 0E
70C5 393 0000'CD
70C8 394
70C8 395 0000'CD
70CB 396
70CB 397 C9
70CC 398
70CC 399
70CC 400
70CC 401
70CC 402
70CC 403
70CC 404
70CC 405
70CC 406
70CC 407
70CC 408
70CC 409
70CC 410
70CC 411
70CC 412
70CC 413
70CC 414
70CC 415
70CC 416
70CC 417

```

N 1

Fiche 1 Frame N1

Sequence 13

```

*****
:
: TEST2F_ERROR THIS ROUTINE IS USED BY TEST 2F TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE
:
*****

```

```

TEST2F_ERR1_2: SET      NA_EXP_REC      ;EXPECTED AND RECEIVED DATA IS N/A
                LDA      EXP_DAT        ;MOVE EXPECTED DATA TO CON_ADD_DAT
                STA      CON_ADD_DAT+3
                CLEAR    NA_OTHER       ;OTHER DATA CONTAINS CURRENT PATTERN
                LXI      H,<^X0000>     ;ERROR MASK IS ZERO
                SHLD     CON_MSK_DAT
                JMP      REPORT_ERR

TEST2F_ERROR:  MVI      A,3             ;ERROR NUMBER 3
                STA      CON_ERR_NUM
                ;
                LDA      EXP_DAT        ;MOVE EXPECTED DATA TO CON_EXP_DAT
                STA      CON_EXP_DAT+3
                ;
                LDA      REC_DAT        ;MOVE RECEIVED DATA TO CON_REC_DAT
                STA      CON_REC_DAT+3
                ;
                SET      NA_OTHER       ;OTHER DATA NOT APPLICABLE
                ;
                LXI      H,<^XFFFF>     ;ERROR MASK INCLUDES ONLY THE LEAST
                SHLD     CON_MSK_DAT    ; SIGNIFICANT BYTE
                LXI      H,<^X00FF>
                SHLD     CON_MSK_DAT+2

REPORT_ERR:    LXI      H,MWCS_A       ;WCS MODULE BEING TESTED
                ;
                LXI      D,CON_MOD_DAT ;LOAD THE MODULE DATA POINT TO BY
                MVI      C,5           ; THE H,L PAIR INTO CON_MOD_DAT.
                CALL     MOVER
                ;
                CALL     CON_ERROR      ;CALL ERROR ROUTINE
                RET

```

```

*****
:
: TEST 2F      APT/RD USART MODEM LOOPBACK TEST
: -----
:
: THE APT/RD USART MODEM LOOPBACK TEST WILL WRITE ALL 0'S,
: ALL 1'S, 0 RIPPLED THROUGH A FIELD OF 1'S, AND 1 RIPPLED THROUGH
: A FIELD OF 0'S TO THE APT/RD USART. IT IS AN EXTENSION OF THE
: USART INTERNAL LOOPBACK SELF TEST TO BE USED FOR APT AND FIELD SERVICE
: PURPOSES. THIS IS AN OPTIONAL TEST, ONLY RUN WHEN EXPLICITLY
: SPECIFIED. FOR THIS TEST TO RUN PROPERLY, EITHER A MODEM OR A
: TURNAROUND CONNECTOR MUST BE CONNECTED TO THE APT/RD USART CABLE.
:
: TEST STEPS:
: 1) SET UP APT/RD USART FOR TEST.
: 2) WAIT FOR XMIT RDY FROM USART.
: 3) ISSUE ERROR 1 OF XMIT RDY NOT SET.
:

```

ZZ-ENKBG-1.0 7000 4

70CC 418  
70CC 419  
70CC 420  
70CC 421  
70CC 422  
70CC 423  
70CC 424  
70CC 425  
70CC 426  
70CC 427  
70CC 428  
70CC 429  
70CC 430  
70CC 431  
70CC 432  
70CC 433  
70CC 434  
70CC 435  
70CC 436  
70CC 437  
70CC 438  
70CC 439  
70CC 440  
70CC 441  
70CC 442  
70CC 443  
70CC 444  
70CC 445  
70CC 446  
70CC 447  
70CC 448  
70CC 449  
70CC 450  
70CC 451  
70CC 452  
70CC 453  
70CC 454  
70CC 455  
70CC 456  
70CC 457  
70CC 458  
70CC 459  
70CC 460  
70CC 461  
70CC 462  
70CC 463  
70CC 464  
70CC 465

0000'CD 2F 3E  
OF 0000'3A  
C9 00D9'D2  
OF 0000'3A  
3D D3 0119'DA

70E2 466  
70E2 467 40 DB  
70E4 468  
70E4 469 37 3E  
70E6 470 43 D3  
70E8 471  
70E8 472 000C'3A  
70EB 473

B 2

Fiche 1 Frame B2

Sequence 14

- 4) WRITE ALL 0'S OUT TO THE USART.
- 5) WAIT FOR REVR DONE FROM USART.
- 6) ISSUE ERROR 2 IF REVR DONE NOT SET.
- 7) READ DATA FROM USART DATA BUFFER.
- 8) ISSUE ERROR 3 IF RECEIVED DATA IS NOT EQUAL TO TRANSMITTED DATA.
- 9) REPEAT STEPS 2-8 FOR DATA OF ALL 1'S.
- 10) REPEAT STEPS 2-8 FOR 0 SHIFTED IN FIELD OF 1'S.
- 11) REPEAT STEPS 2-8 FOR 1 SHIFTED IN FIELD OF 0'S.

ASSUMPTIONS: IT IS ASSUMED THAT EITHER A MODEM WITH WRAPAROUND CAPABILITIES ENABLED OR A TURNAROUND CONNECTOR IS CONNECTED TO THE APT/RD UART.

ERROR 1: XMIT RDY NEVER RECEIVED FROM USART.  
ERROR 2: REVR DONE NEVER RECEIVED FROM USART.  
ERROR 3: USART DATA ERROR.

OTHER: OTHER DATA PRINTED FOR ERRORS 1 AND 2 INDICATES THE DATA PATTERN THAT WOULD BE WRITTEN TO THE USART IF IT WERE PROPERLY ENABLED.

- DEBUG:
- 1) CHECK THAT THE MODEM OR TURNAROUND CONNECTOR IS CORRECTLY INSTALLED AND ENABLED. USE THE ANALOG LOCAL LOOP BACK FEATURE. THE 'TEST' LAMP SHOULD THEN LIGHT. USE THE SINGLE LINE EIA TURNAROUND CONNECTOR, H3248, ATTACHED TO THE APT/RD PORT CABLE.
  - 2) CHECK THAT THE BAUD RATE OF THE MODEM MATCHES THE BAUD RATE OF THE USART. THE USART IS EXTERNALLY CLOCKED AT EITHER 300 OR 1200 BAUD. SWITCH 5 ON WCS E47 SHOULD BE ON FOR 300 BAUD TRANSMISSION, OFF FOR 1200 BAUD TRANSMISSION. A DF03 MODEM OPERATES AT 300 BAUD IF THE 'HS' BUTTON IS OUT, AND 1200 BAUD IF THE 'HS' BUTTON IS IN. THE 'HS' LAMP SHOULD BE LIT IF THE 'HS' BUTTON IS IN, OFF IF OUT. IF A TURNAROUND CONNECTOR IS BEING USED, THE SWITCH SETTING OF E47 SWITCH 5 DOES NOT MATTER.
  - 3) IF THE MACHINE PASSES THE POWER-UP SELFTEST, THEN ERROR 3 INDICATES A PROBLEM WITH THE DRIVERS/RECEIVERS OF THE APT/RD USART (E7 OF THE WCS BOARD). THE INVERTER LOGIC OF PRINTSET WCSN COULD ALSO BE AT FAULT, AS WELL AS THE CABLES AND CONNECTORS TO THE APT/RD PORT.

\*\*\*\*\*  
TEST2F: HEAD <^X2F>

|     |             |                                     |
|-----|-------------|-------------------------------------|
| IN  | TRDB        | :CLEAR OUT USART DATA BUFFER        |
| MVI | A,TRCR_MASK | :SET UP REMOTE CR REGISTER FOR TEST |
| OUT | TRCR        |                                     |
| LDA | ZERO_DAT    | :LOAD THE FIRST DATA PATTERN INTO A |

```

ZZ-ENKBG-1.0 7000 4
70EB 474 000D'CD
70EE 475
70EE 476
70EE 477 000A'3A
70F1 478
70F1 479 000D'CD
70F4 480
70F4 481
70F4 482
70F4 483 000A'3A
70F7 484 0007'32
70FA 485
70FA 486 3C
70FB 487 3C
70FC 488 0008'32
70FF 489
70FF 490 00'07'21
7102 491 AE
7103 492
7103 493 000D'CD
7106 494
7106 495 0008'3A
7109 496 07
710A 497 0008'32
710D 498
710D 499 00FF'D2
7110 500
7110 501 00'07'21
7113 502 34
7114 503
7114 504 00FF'CA
7117 505 3C D3
7119 506
7119 507
7119 508
7119 509
7119 510
7119 511
7119 512
7119 513 0000'CD 51 3E
      OF 0000'3A
      C9 0126'D2
      OF 0000'3A
      3D D3 0135'DA
712F 514
712F 515 OF 3E
7131 516 43 D3
7133 517
7133 518 3C D3
7135 519

```

```

C 2
Fiche 1 Frame C2 Sequence 15
CALL TEST2F_SUB ;CALL ROUTINE TO WRITE AND READ DATA
; FROM APT/RD UART
LDA ONE_DAT ;LOAD A DATA PATTERN OF ALL 1'S INTO A
CALL TEST2F_SUB ;CALL ROUTINE TO WRITE/READ DATA
LDA ONE_DAT ;BACKGROUND OF 1'S SET UP
STA BACKGRND
INR A
INR A
STA DATA_PAT ;FIRST BIT TO TEST
LXI H,BACKGRND
XRA M ;GENERATE SHIFTING PATTERNS
CALL TEST2F_SUB ;CALL WRITE/READ ROUTINE
LDA DATA_PAT ;GET THE PATTERN USED TO GENERATE DATA
RLC ;ROTATE LEFT ONE PLACE
STA DATA_PAT ;MAKE THIS THE NEW PATTERN
JNC 6$ ;KEEP LOOPING FOR ALL 8 BITS
LXI H,BACKGRND
INR M ;INCREMENT E TO ALL ZERO'S
JZ 6$ ;LOOP FOR 1 FLOATING IN FIELD OF 0'S
TAIL
*****
: LAST TEST FOR EVERY CONTOL_SECTION....FORCES END OF SECTION
*****
END_TEST: HEAD <^X51>
MVI A,15 ;RESET UART
OUT TRCR
TAIL

```

ZZ-ENKBG-1.0  
.MAIN.  
Symbol table

Symbol table

D 2  
3-MAY-1982

Fiche 1  
3-MAY-1982 09:58:54  
3-MAY-1982 09:57:25

Frame D2

Sequence 16

VAX-11 Macro V02.46  
WRKDS1:[MAYO.RELEASE]ENKBG.MAC;13 (1)

Page 15

\$PSW = 00000006  
\$SP = 00000006  
A = 00000007  
ALLOWP = 00000003  
APTFLG = 00000004  
AUTO = 00000005  
B = 00000000  
BACKGRND = 00000007 R 02  
BOOTEN = 00000007  
BOOTF = 00000001  
C = 00000001  
CLRACK = 000000A9  
CLRACL = 00000033  
CLRATN = 000000AB  
CLRBSY = 0000002E  
CLRCLK = 00000020  
CLRCSK = 00000024  
CLRCSR = 00000038  
CLRDCL = 00000031  
CLRHLT = 000000AF  
CLRLIT = 0000003C  
CLRMCI = 00000029  
CLRMCK = 0000002B  
CLRPFI = 000000AD  
CLRSS = 00000022  
CLRTIM = 000000A5  
CLRTMR = 0000002C  
CLRUPC = 000000A8  
CLRUPK = 00000026  
CLRWCK = 00000036  
CNT = 000007D5  
CNTA = 00000FA0  
COMPARE \*\*\*\*\* X 02  
CON\_ADD\_DAT \*\*\*\*\* X 02  
CON\_BEGIN\_TEST \*\*\*\*\* X 02  
CON\_ERROR \*\*\*\*\* X 02  
CON\_ERR\_NUM \*\*\*\*\* X 02  
CON\_EXP\_DAT \*\*\*\*\* X 02  
CON\_MOD\_DAT \*\*\*\*\* X 02  
CON\_MSK\_DAT \*\*\*\*\* X 02  
CON\_REC\_DAT \*\*\*\*\* X 02  
CPATTN = 00000083  
CPUACK = 00000082  
CSR07 = 00000085  
CSR15 = 00000086  
CSR23 = 00000087  
D = 00000002  
DATA\_PAT = 00000008 R 02  
DCLO = 00000002  
DISMEM = 000000A7  
DISPAR = 000000A1  
E = 00000003  
ENBMEM = 000000A6  
ENBPAR = 000000A0  
END\_TEST = 00000119 R 02  
ENTRY = 00000000 R 02  
EOS\_FLG \*\*\*\*\* X 02

ERROR\_CON \*\*\*\*\* X 02  
EXP\_DAT = 00000009 R 02  
FILE\_NAME = 00000005 R 02  
H = 00000004  
HEAD = 00000000  
HLTFLG = 00000002  
INITH = 00000035  
INITL = 00000034  
ITCSR = 0000001D  
ITDB = 0000001C  
L = 00000005  
LITERAL = 00000001  
LVL = 00000000  
M = 00000006  
MACSTK1 = 000007D5  
MISC2 = 00000001  
MOVER \*\*\*\*\* X 02  
MWCS\_A \*\*\*\*\* X 02  
NA\_EXP\_REC \*\*\*\*\* X 02  
NA\_OTHER \*\*\*\*\* X 02  
OKV15 = 00000007  
ONE\_DAT = 0000000A R 02  
PARITY\_VECTOR = 00004C22  
POWER\_VECTOR = 00004C25  
READ = 00000080  
READJ1 = 00000003  
READJ2 = 00000004  
REC\_DAT = 0000000B R 02  
REMDIS = 00000006  
REPORT\_ERR = 000000BD R 02  
SETACK = 000000A8  
SETACL = 00000032  
SETATN = 000000AA  
SETBSY = 0000002F  
SETCLK = 00000021  
SETCSK = 00000025  
SETCSR = 00000039  
SETDCL = 00000030  
SETHLT = 000000AE  
SETLIT = 0000003D  
SETMCI = 00000028  
SETMCK = 0000002A  
SETPFI = 000000AC  
SETSS = 00000023  
SETTIM = 000000A4  
SETTMR = 0000002D  
SETUPC = 000000A9  
SETUPK = 00000027  
SETWCK = 00000037  
SKIP\_TEST \*\*\*\*\* X 02  
SYM = 000007D5  
TEST2F = 000000CC R 02  
TEST2F\_ERR1\_2 = 00000083 R 02  
TEST2F\_ERROR = 0000009B R 02  
TEST2F\_SUB = 0000000D R 02  
TRCR = 00000043  
TRCR\_MASK = 00000037

TRDB = 00000040  
TRMODE = 00000042  
TRSR = 00000041  
TRSR\_MASK = 00000002  
TTCR = 0000004B  
TTDB = 00000048  
TTMODE = 0000004A  
TTSR = 00000049  
TUCR = 00000047  
TUDB = 00000044  
TUMODE = 00000046  
TUSR = 00000045  
UPC14 = 00000084  
UPC14A = 00000000  
VERSION = 00000003 R 02  
VNORML = 000000A3  
VSHIFT = 000000A2  
WCSB0 = 00000008  
WCSB1 = 00000009  
WCSB2 = 0000000A  
WRITE = 000000CC  
Y = 00000042  
YBUSRD = 000000EC  
ZERO\_DAT = 0000000C R 02

ZZ-ENKBG-1.0 Psect synopsis  
.MAIN.  
Psect synopsis

E 2  
3-MAY-1982

Fiche 1 Frame E2 Sequence 17  
3-MAY-1982 09:58:54 VAX-11 Macro V02.46 Page 16  
3-MAY-1982 09:57:25 WRKDS1:[MAYO.RELEASE]ENKBG.MAC;13 (1)

+-----+  
! Psect synopsis !  
+-----+

| PSECT name | Allocation       | PSECT No. | Attributes                                              |
|------------|------------------|-----------|---------------------------------------------------------|
| . ABS :    | 00000000 ( 0.)   | 00 ( 0.)  | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| . BLANK :  | 00000000 ( 0.)   | 01 ( 1.)  | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE       |
| CPU6       | 00000135 ( 309.) | 02 ( 2.)  | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE       |

+-----+  
! Performance indicators !  
+-----+

| Phase                  | Page faults | CPU Time    | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| Initialization         | 26          | 00:00:00.12 | 00:00:02.45  |
| Command processing     | 37          | 00:00:00.28 | 00:00:07.95  |
| Pass 1                 | 385         | 00:00:06.17 | 00:01:37.99  |
| Symbol table sort      | 0           | 00:00:00.15 | 00:00:01.09  |
| Pass 2                 | 151         | 00:00:03.06 | 00:00:34.49  |
| Symbol table output    | 14          | 00:00:00.10 | 00:00:00.99  |
| Psect synopsis output  | 2           | 00:00:00.02 | 00:00:00.18  |
| Cross-reference output | 0           | 00:00:00.00 | 00:00:00.00  |
| Assembler run totals   | 618         | 00:00:09.92 | 00:02:25.15  |

The working set limit was 715 pages.

39917 bytes (78 pages) of virtual memory were used to buffer the intermediate code.

There were 10 pages of symbol table space allocated to hold 138 non-local and 11 local symbols.

2474 source lines were read in Pass 1, producing 14 object records in Pass 2.

148 pages of virtual memory were used to define 147 macros.

+-----+  
! Macro library statistics !  
+-----+

| Macro library name                 | Macros defined |
|------------------------------------|----------------|
| SYS\$SYSROOT:[SYSLIB]STARLET.MLB;1 | 0              |

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBG.LIS/OBJECT=ENKBG.OBJ 8085MAC.MAR+MACDEF.MAC+[ ]ENKBG.MAC